AN ATTENTION-BASED MODEL FOR LEARNING DYNAMIC INTERACTION NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

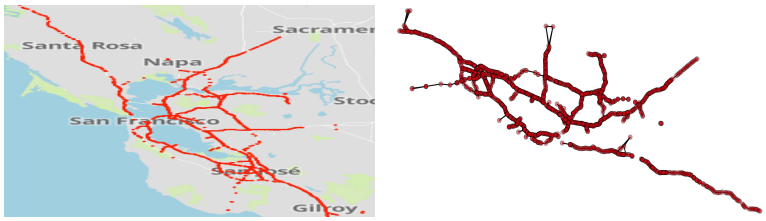
While machine learning models achieve human-comparable performance on sequential data, exploiting structured knowledge is still a challenging problem. Spatio-temporal graphs have been proved to be a useful tool to abstract interaction graphs and previous works exploits carefully designed feed-forward architecture to preserve such structure. We argue to scale such network design to real-world problem, a model needs to automatically learn a meaningful representation of the possible relations. Learning such interaction structure is not trivial: on the one hand, a model has to discover the hidden relations between different problem factors in an unsupervised way; on the other hand, the mined relations have to be interpretable. In this paper, we propose an attention module able to project a graph sub-structure in a fixed size embedding, preserving the influence that the neighbors exert on a given vertex. On a comprehensive evaluation done on real-world as well as toy task, we found our model competitive against strong baselines.

1 INTRODUCTION

In nature, many events are the consequence of the cooperation among different components of a whole system. Similarly, the modern society is characterised by the interaction between different economic and political forces not completely understood. Being able to define such forces is an essential step towards the understanding of the modern world as well as the creation of general intelligence, but it requires the capability of reason at two different level: first, a single individual; second, as a group of interactive units. Moreover, such problem setting is challenging for different reasons: first, a meaningful global structure has to be preserved; then, in large interactive networks some links could be irrelevant, thus introducing noise in the learned embedding.

On the one hand, recurrent neural networks (RNNs) has been proven to successfully solve long-term end-to-end learning tasks Sutskever et al. (2014); Graves & Jaitly (2014); Fragkiadaki et al. (2015); Mikolov et al. (2010). Nonetheless, they cannot condition their distribution based on spatial relations. Recently, the Sequence-to-Sequence model Sutskever et al. (2014) (Seq2Seq) enriched RNNs' expressive power to solve complex sequential tasks. However, one of the most significant limitations of such systems lies in its capability of handling only array data structure, while many different problems are better expressed as a graph structure. On the other hand, traditional graph embedding techniques Perozzi et al. (2014); Cavallari et al. (2017); Tang et al. (2015) presuppose the existence of a well-defined graph structure, while usually these models are not suited to preserve the evolution of the node's attributes Dong et al. (2017); Wei et al. (2017) or they are not able to automatically determine the influence that the neighbours exert on a target node Ying et al. (2018); Battaglia et al. (2016) or do not leverage the supervised information during learning Perozzi et al. (2014); Tang et al. (2015); Grover & Leskovec (2016).

Jointly learn an interactive graph structure and predicting the future behaviour of the system is a non-trivial task. A straightforward solution is to compute a pairwise influence between each pair of nodes in the dataset. Such computed scores can be used as weights of a fully connected graph. Then any inference algorithm can be used to perform regression or classification task at the node level. However, such an approach is unscalable and lack a global view of the problem since each pair of nodes is evaluated independently. Nevertheless, such problem setting is a typical scenario in many real-world applications. For example, consider that it is possible to represent the traffic flow of a given geographic area through a network structure of interconnected traffic stations where the



(a) Traffic stations present in the bay area

(b) Interaction graph between different stations

Figure 1: As motivational example Fig. 1b shows the latent interaction graph formed by the different traffic monitor stations around the San Francisco bay area located according to Fig. 1a.

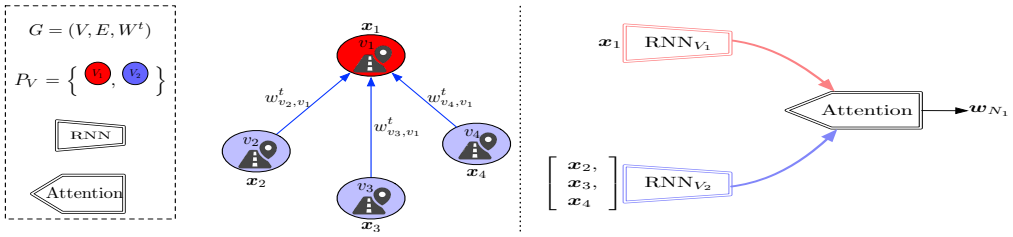
node's attributes are proportional to the car flow across such stations (Fig. 1). The inference model then has to reason over a multitude of connected stations in order to predict the correct flow, since it is not a single car that generates traffic jam, but it is a multitude of vehicles at close locations that cause traffic congestion. Having such an intelligent system would be valuable to most of the transportation authorities and urban planner willing to predict and manage the road's bottleneck and improve the urban design.

During the last years, researchers found useful to represent such interactive structure with a graph, more in detail, spatio-temporal graphs (st-graphs) Li & Nevatia (2008); Lezama et al. (2011); George & Shekhar (2008); Liu et al. (2011) became a powerful tool to represent and study such systems. In Jain et al. (2016), the authors propose a pioneering framework, named Structural-RNN, where the temporal modeling ability of RNNs is combined with st-graphs generating a mixture of RNNs able to combine the best of both worlds. Recently, interaction networks Battaglia et al. (2016); van Steenkiste et al. (2018); Sukhbaatar et al. (2016) became a popular framework to solve physical and structured problems. Exploiting a similar design to well-known GNNs Scarselli et al. (2009), where messages are passed over connected graphs, these networks can efficiently separate the problem's objects from their relations; thus adapting at the changes expected in such dynamic environment. However, most of these models adopt a pairwise approach which is not able to properly preserve the graph structure Santoro et al. (2017); van Steenkiste et al. (2018) or they are not able to capture the evolution of the entire systemKipf et al. (2018); Hoshen (2017).

In this work, we propose a novel *attention-based interaction network* able to leverage a dynamic interactive structure representative of the underlying problem. Similarly to Veličković et al. (2017), our model employs a self-attention mechanism to evaluate the influence that all neighbours have *w.r.t.* a given vertex. However, we are seeking for a fixed size embedding representative of the evolving interactive structure as well as of the node attributes to perform a regressive or auto-regressive task at the node level. Note that we are also different from Lee et al. (2018) since we propose a supervised learning algorithm trained end-to-end to perform a node-level task, instead of a reinforcement learning formulation for graph-level classification. We extensively evaluate such an approach on three real-world datasets as well as on three synthetic datasets. The main contributions of this paper are: 1. we propose a model able to automatically infer the interactive structure of a problem; 2. to better preserve the problem's semantic, we propose a different parametrisation of the node and neighbours; 3. we show that our attention architecture is comparable or outperforms structured and unstructured algorithms.

2 Dynamic Interaction Attention Network

At each instant a node is influenced by its internal state as well as by the dynamic interaction with its neighbours. To this end, the objective of this work is to propose an architecture able to automatically learn a dynamic graph structure $G = (V, E, W^t)$, where V is the nodes' set, E is the edges' set and W^t is the set of the edge's weight at time t ($w^t_{j,i}$) representative of the influence exerted by node v_j on node v_i. On the one hand, to grasp such dynamic graph structure, it is necessary to learn appropriate edge's weights based on: 1. the node's attributes; 2. the neighbourhood behaviours; 3. and the system performance at previous time-steps. For example, consider the traffic prediction problem: in order to obtain an accurate prediction of the car's flow through a target station, it is



(a) Interaction graph between different problem objects

(b) Parameter sharing between semantically similar edges

Figure 2: Graphical representation of our interaction graph, where semantically similar relations share the same parameters and the attention model is used to learn the factor interactions.

necessary to also consider the status of the neighbour stations. On the other hand, with respect to Lee et al. (2018) our objective is to perform a regression task at the node level. To better preserve the graph structure, we partition the nodes based on the problem's aspects they represent, formally we distinguish between:

$$P_V = \left\{ \begin{array}{l} \text{the target nodes} \rightarrow V_1 \\ \text{the neighbours nodes} \rightarrow V_2 \end{array} \right\}$$

Such partition P_V aims to better preserve the difference between node's features *w.r.t.* the neighbours' attributes. Let us define a vertex i as v_i, similarly let us define the attributes time series of v_i as $\boldsymbol{x}_i = [\boldsymbol{x}_i^1, ..., \boldsymbol{x}_i^T]$ where $\boldsymbol{x}_i^t \in \mathbb{R}^d$ represent the feature vector at time t. Then, Fig. 2 shows a graphical representation of the node partitioning. Specifically, Fig. 2a report a target node v_1 with three neighbours $[v_2, v_3, v_4]$. According to our goal, at time t, we aim to learn the target node feature status ($\boldsymbol{x}_1^{t+1}$) subject to the neighbours' influence and attribute represented respectively by the edge weight $w_{2,1}^t, w_{3,1}^t, w_{4,1}^t$ and $\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4$. Thus, in Fig. 2b we obtain two distinct parametrisation: the first one for the node features ($\text{RNN}_{V_1}(\boldsymbol{x}_1)$), and the second representative of the neighbours attributes ($\text{RNN}_{V_2}(\begin{bmatrix} \boldsymbol{x}_2, \\ \boldsymbol{x}_3, \\ \boldsymbol{x}_4 \end{bmatrix})$). Note that in Fig. 2b $\begin{bmatrix} \cdot, \\ \cdot \end{bmatrix}$ stand for the concurrent processing of different neighbours in the same batch, also note that two distinct RNN are used to encode the different time series. Such partitioning enables us to reduce the number of parameters to learn since all the neighbours share the same parametrisation while it better preserve the problem's semantic since the nodes and the respective neighbours have different encoding procedure. We define N_i as the set of the neighborhood of node v_i; thus the propose attention mechanism aims to learn a meaningful edge weights $\boldsymbol{w}_{N_1} = [w_{2,1}^1, w_{3,1}^1, w_{4,1}^1, ..., w_{2,1}^T, w_{3,1}^T, w_{4,1}^T]$ for all the time-step and for all the neighbours of the target vertex v_1.

To this end, the proposed *Dynamic Interaction Attention Network* (DIAN) is technically different from Hoshen (2017); Veličković et al. (2017) since it uses a different attention formulation that not only span over the graph structure but also over time; moreover, it is based on a different edge representation.

Note that the proposed DIAN model is different from the Structural-RNN architecture because it does not provide a prior interaction structure. Instead, it automatically learns the influence $w_{j,i}^t$ that each edge exert on a given incident node. Such automatic mechanism allows the DIAN architecture to better generalise on new tasks as it does not require the manual crafting of each relation. As a side effect, due to the attention module used, the interaction structure learned becomes explicit and straightforward to interpret. Our model is also different from NRI Kipf et al. (2018) since we do not process each edge independently through a message passing process, but we obtain a global view of the problem through the attention mechanism. Also note that partitioning the edges in semantically different group reduce the number of parameters to learn *w.r.t.* the NRI and the Structural-RNN architecture, since DIAN do not have a different network for each edge type, but only different encoders for the a given node and its neighbours.

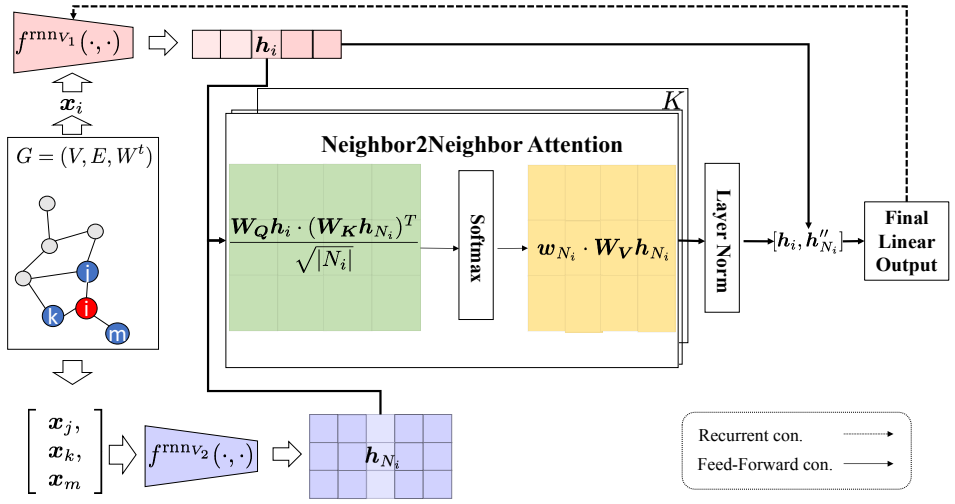


Figure 3: Overall model architecture.

2.1 Model Architecture

Most of the related work firstly encode the graph's edges as concatenation of the nodes features and then apply a graph attention network (GAT) to learn a proper network structure Duan et al. (2017); Hoshen (2017); Veličković et al. (2017); van Steenkiste et al. (2018) or process each problem aspects with different RNNs organised in a feedforward structure Jain et al. (2016). However, the first approach is not able to accurately differentiate between the node attributes as well as the neighbours' attribute, while the second approach is not able to automatically learn the influence of each neighbour.

To preserve the best of both worlds, we propose to obtain different parametrisation for the node and the neighbours' aspects. Since the nodes' state change over time, we have to ensure that such dynamism is preserved. RNNs are a deep learning model that has been proven capable to preserving the smooth transition between subsequent time-step of a given sequence Graves & Jaitly (2014); Fragkiadaki et al. (2015); Mikolov et al. (2010), thus are an excellent architecture to capture such dynamics. An RNN hidden state, relative to the input vertex i, can be expressed as:

$$\boldsymbol{h}_i^t = f^{\text{rnn}}(\boldsymbol{x}_i^t, \boldsymbol{h}_i^{t-1})$$

where $f^{\text{rnn}}(\cdot,\cdot)$ is a non-linear function that combines two source of information. Note that in this work we used the well-known GRU model Cho et al. (2014) as recurrent function $f^{\text{rnn}}(\cdot,\cdot)$.

As shown in Fig. 3, given a target node v_i, we obtain a different parametrisation of the node and neighbours attributes using different RNNs. Specifically, we define $\boldsymbol{h}_i^t = f^{\text{rnn}_{V_1}}(\boldsymbol{x}_i^t, \boldsymbol{h}_i^{t-1})$ as the target node representation at time t and $\boldsymbol{h}_{N_i}^t = f^{\text{rnn}_{V_2}}([\boldsymbol{x}_j^t]_{j \in N_i}, \boldsymbol{h}_{N_i}^{t-1})$ as the neighbours representation at time t, where in this case $[\cdot]_{j \in N_i}$, as show in Fig. 2b, stand for a batch composed by all the j neighbours present in the set N_i. Consequently we define $\boldsymbol{h}_i = [\boldsymbol{h}_i^1, ..., \boldsymbol{h}_i^T]$ as the encoded time series of vertex i processed by $f^{\text{rnn}_{V_1}}$ and $\boldsymbol{h}_{N_i} = [\boldsymbol{h}_{N_i}^1, ..., \boldsymbol{h}_{N_i}^T]$ as the time series of v_i's neighbours processed by $f^{\text{rnn}_{V_2}}$. We are aware that in this setting $f^{\text{rnn}_{V_2}}$ is trained more times than $f^{\text{rnn}_{V_1}}$ due to the different batch size: approximately D times more where D is the average degree of the graph G. Although this raises some theoretical concern, empirically we have not found it to be an issue.

In order to obtain the dynamic weight representative of the influence of each neighbour, a self-attention mechanism is employed. Specifically, we propose to use the *scaled dot product attention* Vaswani et al. (2017) to learn the edges weight:

$$\boldsymbol{w}_{N_i} = softmax\left(\frac{\boldsymbol{QK}^T}{\sqrt{|N_i|}}\right) \tag{1}$$

$$= softmax\left(\frac{\boldsymbol{W_Q h}_i \cdot (\boldsymbol{W_K h}_{N_i})^T}{\sqrt{|N_i|}}\right) \tag{2}$$

where $\boldsymbol{w}_{N_i} = [\boldsymbol{w}_{j,i}]_{j \in N_i}$, and $\boldsymbol{W_Q}, \boldsymbol{W_K}$ are linear transformation to ensure that $\boldsymbol{h}_i$ and $\boldsymbol{h}_{N_i}$ belong to the same semantic space. Respect to Veličković et al. (2017) we are different in two aspects: first, we jointly attend both over time and overall the neighbours; secondly, we do not attend directly on the edges representation as in Veličković et al. (2017), but rather on the nodes their-self.

The weights are then used to obtain a relevant fixed-size embedding of the meaningful neighbours:

$$\boldsymbol{h}'_{N_i} = \boldsymbol{w}_{N_i} \cdot \boldsymbol{W_V} \boldsymbol{h}_{N_i} \tag{3}$$

where $\boldsymbol{W_V}$ is an other linear transformation. Such formulation enables the DIAN architecture to focus only on the relevant sub-graph, reducing the noise present in the structure of complex systems. Note, at time t' the influence of a neighbour j over a node i is defined as :

$$\boldsymbol{h}'^{t'}_j = \sum_{1 \leq t \leq t'} w^t_{j,i} \cdot \boldsymbol{W_V} \boldsymbol{h}^t_j \tag{4}$$

thus, we not only focus on the relevant neighbours, but we also align the time series of each neighbours *w.r.t.* the target node.

To increase the expressive power of our model K parallel attention heads are executed as suggested in Veličković et al. (2017); Vaswani et al. (2017), but instead of using an average polling function, we found more useful to linearly combine (trough $\boldsymbol{W_H}$) each attention head to find the final representation:

$$\boldsymbol{h}''_{N_i} = \sigma(\boldsymbol{W_H}[\boldsymbol{h}'^k_{N_i}]^K_{k=1}) \tag{5}$$

where $\boldsymbol{h}'^k_{N_i}$ stand for the neighbours representation of the k-th attention head, $\sigma(\cdot)$ as an activation function (we used ReLU) and $[\cdot]^K_{k=1}$ stand to the concatenation of all the K attention output. Note that we also have found empirically useful to apply a layer normalisation Ba et al. (2016b) on the final neighbours embedding.

The final sub-structure embedding relative to a target vertex v_i is then obtained as the concatenation $[\boldsymbol{h}_i, \boldsymbol{h}''_{N_i}]$ and it is used to perform the regression task unifying the node's information with the relevant neighbours. Also, we have found useful to add recurrent connection from the final output to the target node input, such connection have been proven effective when modeling volatile time-series. It should be noted that:

1. Since we deal with dynamic influence, at time t, it is important to mask all the $t+1, ..., T$ time-steps in order to maintain the time causality of the problem.
2. Unlike GAT Veličković et al. (2017), our network architecture has to jointly attend on different neighbours as well as on different time-steps. Moreover, we present a different attention mechanism as well as a different node representation able to better preserve the difference between the node's attributes and neighbours' attributes.
3. As stated in Veličković et al. (2017), such attention architectures are not able to scale on interaction graphs with a high average degree. One possible solution is to develop a linear attention formulation Ba et al. (2016a) or to follow the recent trend in GCN Chen et al. (2018); Hamilton et al. (2017); Ying et al. (2018) where only a sample of the total neighbours is processed at each batch; however this is left as future works.

3 EXPERIMENTS

To evaluate the insight of Sec. 1 and 2, we have designed a series of real-world as well as toy tasks. The designed toy tasks are synthetically generated dataset created with the specific purpose of studying the convergence of the DIAN model w.r.t. some specific data characteristics. Instead, the real-world datasets assess the performance of the proposed architecture on existing problems. Note that during all the experiments we used the mean squared error (MSE) as loss function and the rooted mean squared error (RMSE) as evaluation metrics.

3.1 DATASETS

Synthetic Datasets. The synthetic datasets have been designed to replicate the same problem setting of the real-world regression task while challenging three main abilities required by the proposed

architecture. Moreover, each synthetic dataset is composed of 10K examples for training, 2K for evaluating and 2K for testing.

• The *Dynamic Noise dataset* aims to test the dynamic filtering ability of our neighbours attention mechanism. Specifically, a monotonically increasing sequence is assigned to each vertex. Each target node is then connected with a random number of neighbours ranging between one and four. While an irregular behaviour characterises one neighbour, the other neighbours have a time series correlated to the one of the target node. At each time-step, the goal is to compute the sum between the values of the target node and the relevant neighbours. Whit this dataset, we aim to test the attention representation ability while having a variable number of relevant neighbours.

• The *Time-shifted dataset* aims to test the aligning time series ability of our model *w.r.t.* similar architecture such as GAT and structural-RNN. Specifically, each target node is associated with a triangular wave-formed time series, while each neighbour has a relevant wave-formed series shifted ahead in time and amplitude. The objective is to predict the one-step-ahead value of the target node. In order to solve this task, a given model has to align the node time series with its neighbours to detect the shift point of the triangular wave-form sequences correctly.

• The *Noise Time-shifted dataset* combines the two previous task to investigate which network architecture is able to align and detect a variable number of relevant neighbours jointly. To this purpose, this synthetic dataset presents a target nodes characterised by a triangular wave-form time series, up to two irrelevant neighbours and at least two relevant neighbours having the same node's triangular time series but shifted in time and in amplitude by a random offset. Note also that in this task the final objective is to predict the node's next step value.
An example for each handcrafted datasets is reported in Appx. 4.1.

Real-World Datasets. Three real-world datasets are used to evaluate the proposed architecture. Please note that while in Sec. 1 we use the traffic flow prediction problem as a motivational example, here we also test our model on different tasks such as energy load forecast and credit risk. We argue that all those tasks have the same problem structure. For example, buildings closely geo-located are subject to the same climatic factors; thus they share a common consumption pattern. Similarly, in the credit risk context, the company owner and the company manager risk is related to the company credit status.

• The *EnerNOC GreenButton*[1] *dataset* contains anonymized geo-located energy usage data from 100 commercial/industrial sites over the entire 2012. The energy consumption is re-sampled at 3-hour intervals, anomalies readings are eliminated, and the energy consumption is divided by the building's square footage to have a more comparable metric. Note that, all the time series are normalized through the softplus function ($softplus(x) = log(1 + e^x)$) and categorical features as the building type, the time of day and the day of the week are encoded in a one-hot format. We assume that buildings closely geo-located are subject to the same climatic factors, so they share a common consumption pattern. To this end, top-4 geographically closest sites are used as neighbours of a given building.

• The *PeMS Bay-Area traffic*[2] *dataset*, as shown in Fig. 1a, consists of traffic data crawled from 3665 traffic-control stations around the San Francisco Bay area. Traffic data consist of car flow (car/10 min.) and average speed (mph) through each station for the weekdays ranging from May 2017 to July 2017. Each training example is formed by one-day data for a given station, and the objective is to predict the one hour ahead traffic congestion. We found useful to normalize the time series with a scaled softlog function ($softlog(x) = log(1 + \frac{x}{10})$). Our intuition is that geographically closely stations suffer from similar traffic condition, so for each traffic-control terminal we compute top-6 closest stations as potential neighbours.

• The *Bank risk dataset* consists of a private dataset of bank customers. Since Basel 2 agreement, banks are required to wisely execute a credit risk program to better handle loans and premiums discounts. Thus, for each customer, the internal credit status, the internal risk, the balance sheet and socio-demographic information are monthly provided. Moreover, a special section also identifies the links that bind customers with each other: for example, a family account can be the guarantee of a son's account, or the company owner and the company manager are related to the company credit

[1] `https://open-enernoc-data.s3.amazonaws.com/anon/index.html`
[2] `https://bit.ly/2sB27Pa`

Table 1: Datasets summary. Note that for all the datasets, all the examples related to a given node and associated neighbors are only present in the training, evaluation or test set, thus we are in an inductive learning setting.

	EnerNOC	**PeMS**	**Bank**	**Synthetic**
Node number	100	3665	13953	14000
Neighbors for each node	4	6	4	4
Sequence lenght	16	138	17	10
Training examples	9570	50000	9676	10000
Eval examples	4176	12000	2000	2000
Test examples	4002	11300	2000	2000

status. Our intuition is that exploiting such relations could enhance the evaluation of the future risk of each customer. The goal of this dataset is to predict the risk score one time-step ahead. The overall datasets characteristics are summarized in Tab. 1.

3.2 BASELINES

We designed our baselines to specifically evaluate how different neural networks can learn and exploit structured information in the data.

• *Structural-RNN* Jain et al. (2016): directly model a graph structure given in input thanks to its feed-forward architecture, but is not explicitly able to dynamically adapt to the changes is such graph structure.

• *Simple-RNN* Cho et al. (2014): to investigate the impact that the neighbours' information has on the final performance, we use a single GRU to only model the node sequences.

• *Structural-FF*: to evaluate the importance of modeling the smooth transition present in our time series, we propose a baseline that uses the same feed-forward architecture of the S-RNN architecture, but it employees multilayer perceptrons (MLPs) instead of the RNNs.

• *Simple-FF*: to understand how difficult it is to solve the proposed tasks, a simple MLP that only process the node information is used in the evaluation. Note that such network architecture, can neither model the smooth time transition, neither the structured information.

• *GAT* Veličković et al. (2017): to evaluate the impact of our edge formulation and the proposed node partitioning, we also use GAT as a baseline. While being an architecture similar to the proposed one, the GAT is not able to preserve the smooth time translation since process each time-step independently.

• *RNNGAT*: since the used datasets are all time dependent; we decided to enhance the GAT using an RNN instead of the MLP to encode the edges and preserve the time transition better. Note that also in this architecture the well-known GRU design is used as the RNN implementation.

Moreover, we designed degenerated versions of our model to evaluate the impact of our proposed edge encoding, the recurrent connection and the attention module:

• *DIAN*^{rec}: is a degenerated version the model shown in Fig. 3 that do not present the recurrent connection.

• *DIAN*^{trans}: extend the findings of the work done in Vaswani et al. (2017) to the graph embedding setting. In this architecture, each edge is encoded using MLPs instead of RNNs. Then the attention architecture is expected to jointly learn the graph architecture as well as the time dependencies.

3.3 EXPERIMENTS SETUP

We report a short summary of the parameters used during the experiments in Tab. 2. To obtain a fair comparison among all the baselines, we set the same number of hidden units u in all the evaluated models. In all the synthetic datasets, in particular, we set $u = 5$ while in the real-world forecasting problems we set $u = 128$. We have used 4 attention heads ($h = 4$) in all the experiments since the results shown in Fig. 4a demonstrate that 4-heads give the best trade-off between the network

Table 2: Parameters setting used during the experiments.

	Symbol	Synthetic dataset	Real-world datasets
Hidden units	u	5	128
Attention heads	h	4	4
Learning rate	lr	0.01	0.01

Table 3: Results on synthetic datasets.

Model	Dynamic Noise	Time-shifted	Noise Time-shifted
	RMSE	*RMSE*	*RMSE*
Structural-RNN	1.56	0.20	0.41
Simple-RNN	5.39	0.53	0.54
Structural-FF	2.21	0.63	1.19
Simple-FF	5.64	1.46	1.44
DIAN	**0.51**	0.17	**0.23**
DIAN$^{\text{rec}}$	0.90	**0.16**	0.24
DIAN$^{\text{trans}}$	5.39	0.29	0.35
GAT	4.36	0.60	1.23
RNNGAT	1.21	0.25	0.4

complexity and the performance improvements. Adagrad Duchi et al. (2011) is used as optimizer with 0.01 as learning rate ($lr = 0.01$).

Moreover, it is worth noting that for all the experiments a ReLU activation function is applied after the encoding functions $f^{\text{rnn}}()$, while in the real-world dataset, an ELU activation function is also applied after the final linear projection layer. It is also worth mentioning that, for the PeMS dataset, a temporal window (tw) is used to reducing the number of time-steps considered during back-propagation and speed up the learning process of the different models. In particular, on the PeMS dataset, we use a window size of 20 time-steps ($tw = 20$). Finally, note that all the models are trained for 100 epochs while the evaluation is performed every ten epochs. The parameter setting that better perform on the eval set is then used for testing. **Remark**: The reported results are the average of four independent execution.

3.4 Results on synthetic datasets

Tab. 3 report the test results on the synthetic datasets for all the architecture mentioned in Sec. 3.2. As it is possible to see, the proposed DIAN model can outperform all the baselines obtaining a relative improvement between 4.3% and 76.5% in two out of three tasks, but it is outperformed by the DIAN$^{\text{rec}}$ architecture by 6.2%. However, such experiments demonstrate the advantages of the proposed node's parametrisation and attention mechanism over a feed-forward architecture or a GAT network. In details, *w.r.t.* the Structural-RNN our model can automatically focus only on the relevant neighbours; while *w.r.t.* the GAT model, the proposed architecture can better model the smooth temporal evolution of the environment. Instead, the RNNGAT results validate the idea of having a different parametrisation for the node and the neighbours, since the RNNGAT model is outperformed by the proposed DIAN architecture, especially on the Dynamic Noise and the Noise Time-shifted datasets. On the one hand, adequately encoding the edges is an important factor, since the performances of the DIAN$^{\text{trans}}$ and GAT model are significantly lower that others models. On the other hand, DIAN$^{\text{rec}}$ is a valuable baseline, providing comparable results to the DIAN model. However, the Dynamic Noise results demonstrate that the recurrent connection effectively helps to model volatile time series. Additional explanations are reported in Appx. 4.1.

3.5 Results on real-world datasets

In Tab. 4, we report the results of the experiments conducted on the three real-world datasets. As shown, the proposed DIAN architecture outperforms all the baselines obtaining relative improvements that range form 0.76% (Structural-RNN and DIAN$^{\text{rec}}$ in the bank dataset) to 803.79% (DIAN$^{\text{trans}}$ in the PeMS dataset).

Table 4: Real-world datasets' results.

Model	**EnerNOC**	**PeMS**	**Bank**
	RMSE	*RMSE*	*RMSE*
Structural-RNN	0.045	55.85	1.33
Simple-RNN	0.049	58.75	1.36
Structural-FF	0.067	65.91	1.44
Simple-FF	0.063	71.4	1.45
DIAN	**0.028**	**38.86**	**1.32**
DIAN$^{\text{rec}}$	0.046	55.65	1.33
DIAN$^{\text{trans}}$	0.055	347.87	4.43
GAT	0.061	251.51	1.37
RNNGAT	0.055	245.95	1.36

On the one hand, it is interesting to observe how the EnerNOC dataset presents a valuable interactive structure since the Structural-RNN method performs comparably to the DIAN$^{\text{rec}}$. On the other hand, the DIAN model is able to outperform all the others methods significantly. This suggests that the recurrent connection also provide valuable information when learning the interactive structure. Thus, the DIAN model can obtain a better graph representation.

Similarly, the PeMS dataset presents a strong interactive structure since the DIAN$^{\text{rec}}$ can outperform baseline such as the Simple-RNN by a relative 5.2%. Moreover, the proposed DIAN model is able to better align and learn a valuable interactive structure since outperform both the Structural-RNN and the DIAN$^{\text{rec}}$ architectures. Notice that, neither the DIAN$^{\text{trans}}$ neither the GAT neither the RNNGAT can learn a meaningful problem representation. This could be a consequence of their design which it is not optimised to learn a smooth transition of the dynamic interactive structure; recall that the attention mechanism of the RNNGAT does not span over subsequent time-steps, so producing sub-optimal results. Finally, note that such results validate on real-world problems the hypothesis of having a different parametrisation for each problem aspect to better preserve the graph structure since the RNNGAT performances are significantly worsted that the DIAN and the Structural-RNN approach on EnerNOC and PeMS.

Instead, the Bank dataset seems to present an unclear interactive structure, since complex models, such as the DIAN or the Structural-RNN are comparable to a Simple-RNN (2.66% of relative improvement).

Lastly, as for the toy datasets, we report additional information in Appx. 4.2.

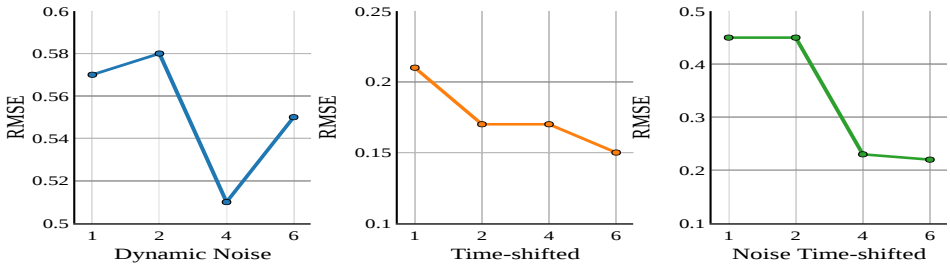
3.6 Sensitivity

To better evaluate the proposed model, we tested the impact of adding additional attention heads in Fig. 4a. Such experiments validate the idea that adding additional attention heads can focus on multiple aspects of the problem and create a richer embedding. However, on the toy task, four heads provide the best trade-off between the model complexity and the performances. For example, $h = 4$ is not suffering from over-fitting in the Dynamic Noise dataset and present comparable performances on the Noise Time-shifted dataset.

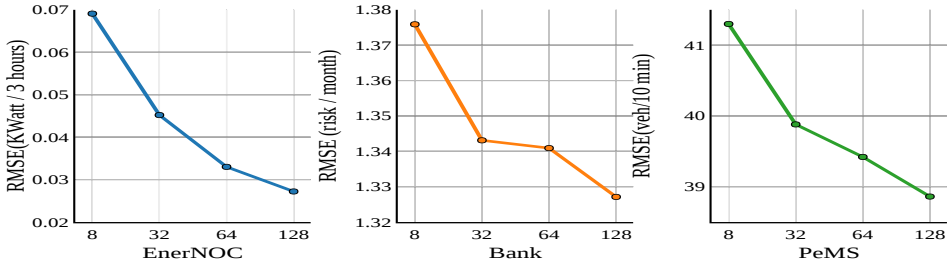
Finally, Fig. 4b shows the impact that the hidden dimension (parameter u) has on the real-world performances. As expected, a bigger hidden layer provides better performances. However, $u = 128$ provide significant better performances *w.r.t.* other settings.

4 Conclusion

In this paper, we proposed a novel dynamic graph attention network. Unlike previous works, our model: 1. can jointly learn the underlying graph and align the nodes' time series; 2. gives a human-readable interpretation of the actual underlying graph structure; 3. can easily adapt to graph with different neighbours number, but cannot handle interaction graph with high degree. While this is not a problem for the tested tasks, further work is required in order to relax this assumption and, hence, develop linear attention. Note that a linear attention formulation would be highly valuable



(a) Impact of the number of attention head on toy datasets



(b) Impact of hidden dimension on real-world datasets

not only in the context of GNN but also in reinforcement learning. An alternative solution is to develop a neighbours sampling process as proposed in Chen et al. (2018); Hamilton et al. (2017), but such alternative is left as future work.

Whit respect to the comparable GAT architecture, the DIAN model is able to preserve the graph structure better, thanks to the node partitioning; compared to the GCN, the proposed model can give different importance score to the neighbours, thus better model the interactive structure.

REFERENCES

Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using Fast Weights to Attend to the Recent Past. *arXiv:1610.06258 [cs, stat]*, October 2016a. arXiv: 1610.06258.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016b.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and others. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pp. 4502–4510, 2016.

Sandro Cavallari, Vincent W. Zheng, HongYun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *CIKM*, pp. 377–386, 2017.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, pp. 135–144, 2017.

Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-Shot Imitation Learning. *arXiv:1703.07326 [cs]*, March 2017. arXiv: 1703.07326.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pp. 4346–4354. IEEE, 2015.

Betsy George and Shashi Shekhar. Time-aggregated graphs for modeling spatio-temporal networks. In *Journal on Data Semantics XI*, pp. 191–212. Springer, 2008.

Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pp. 1764–1772, 2014.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Yedid Hoshen. VAIN: Attentional Multi-agent Predictive Modeling. *arXiv:1706.06122 [cs]*, June 2017.

Ashesh Jain, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-RNN: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5308–5317, 2016.

Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. *CoRR*, abs/1802.04687, 2018.

John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1666–1674. ACM, 2018.

José Lezama, Karteek Alahari, Josef Sivic, and Ivan Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 3369–3376. IEEE, 2011.

Yuan Li and Ram Nevatia. Key object driven multi-category object recognition, localization and tracking using spatio-temporal context. In *ECCV (4)*, pp. 409–422, 2008.

Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1010–1018. ACM, 2011.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, pp. 701–710, 2014.

Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*, 2017.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *Trans. Neur. Netw.*, 20(1):61–80, January 2009. ISSN 1045-9227. doi: 10.1109/TNN.2008.2005605.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2244–2252, 2016.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, pp. 1067–1077, 2015.

Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *CoRR*, abs/1802.10353, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 6000–6010, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

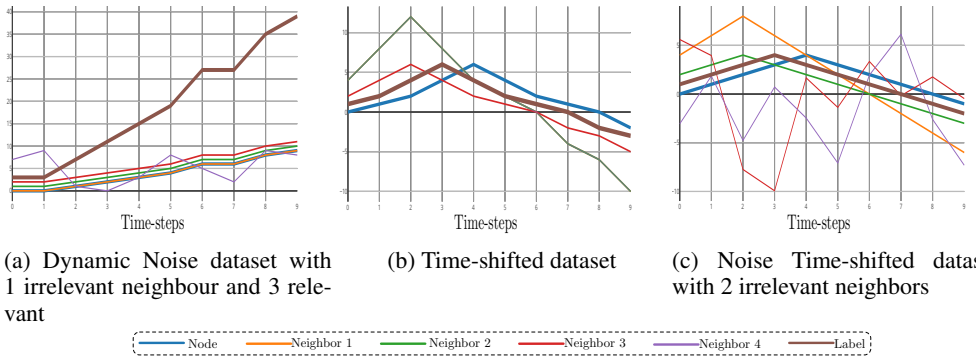
Xiaokai Wei, Linchuan Xu, Bokai Cao, and Philip S Yu. Cross view link prediction by learning noise-resilient representation consensus. In *Proceedings of the 26th International Conference on World Wide Web*, pp. 1611–1619. International World Wide Web Conferences Steering Committee, 2017.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. *arXiv preprint arXiv:1806.01973*, 2018.

APPENDIX

4.1 SYNTHETIC DATASET

Dataset Training Instance Fig. 5 report an example of the three synthetic dataset generated to validate the proposed DIAN architecture. Please note that the target time series of the Dynamic Noise dataset (Fig. 5a) has a significantly bigger dynamic; thus require the model to adapt to sudden changes.

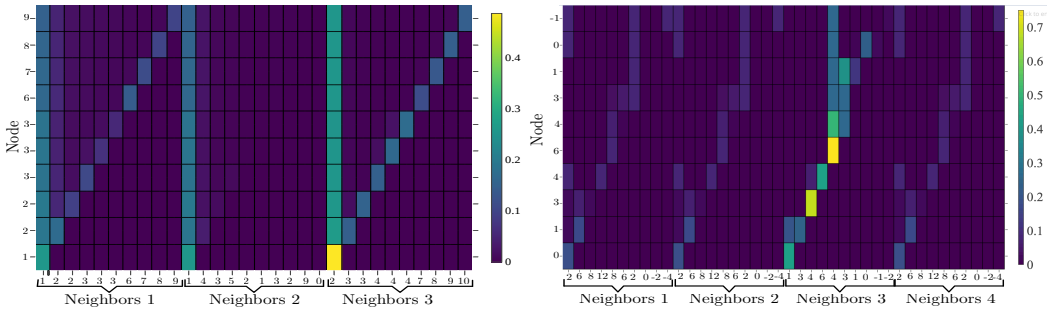


(a) Dynamic Noise dataset with 1 irrelevant neighbour and 3 relevant

(b) Time-shifted dataset

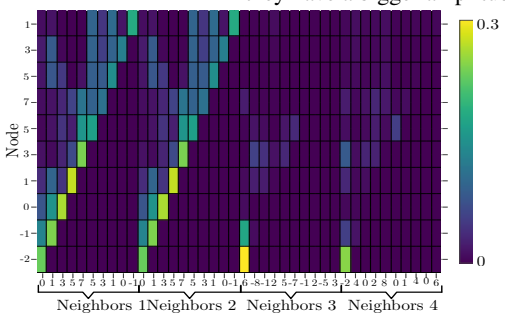
(c) Noise Time-shifted dataset with 2 irrelevant neighbors

Figure 5: A training example for each synthetic dataset. Note that the neighbors are represented by a thinner line w.r.t to the node and label time series. Moreover, noise neighbors have the thinnest line.



(a) Dynamic Noise attention output: Note that this example present a structure with only three neighbours

(b) Time-shifted attention output: Note that the time-series of neighbour 1, 2 and 4 are downsampled, since they have a bigger amplitude.



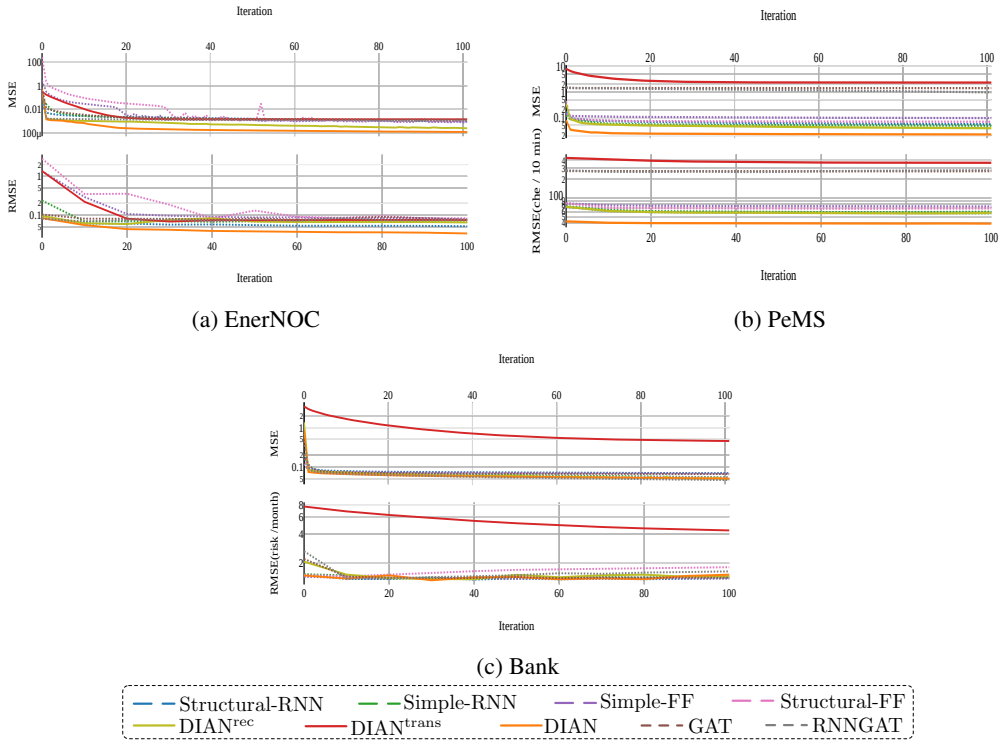
(c) Noise Time-shifted: Note that neighbor_3 and neighbor_4 are irrelevant

Figure 6: Attention output for the DIAN model on the different toy tasks. Note that on the x-axes are reported the neighbours time series $(\boldsymbol{x}_j \mid j \in N_i)$ while the y-axes report the node time series $(\boldsymbol{x}_i)$.

Attention Output As shown in Fig. 6, thanks to the attention mechanism, we can investigate the behaviours of our models. For example, Fig. 6a shows how the DIAN architecture can correctly shadow the first edge due to its noise behaviour. Also, it is possible to see how the model adapt to neighbourhoods of different size since in this example there are only three neighbours up to a maximum of four. Fig. 6b and 6c instead prove that the propose architecture can jointly align and detect irrelevant incoming edges, while correctly detect the shift-point in the remaining relevant edges. Note that for the Time-shifted dataset, the model is also able to down-sample the time series with a bigger amplitude.

4.2 REAL-WORLD DATASETS

Training and Evaluation Performances Fig. 7 report the training and evaluation performances for all the models on the three real-world datasets. Consistently with the results of Tab. 4, it is possible to notice how the DIAN architecture consistently achieve a faster training as well as a lower error.



(a) EnerNOC

(b) PeMS

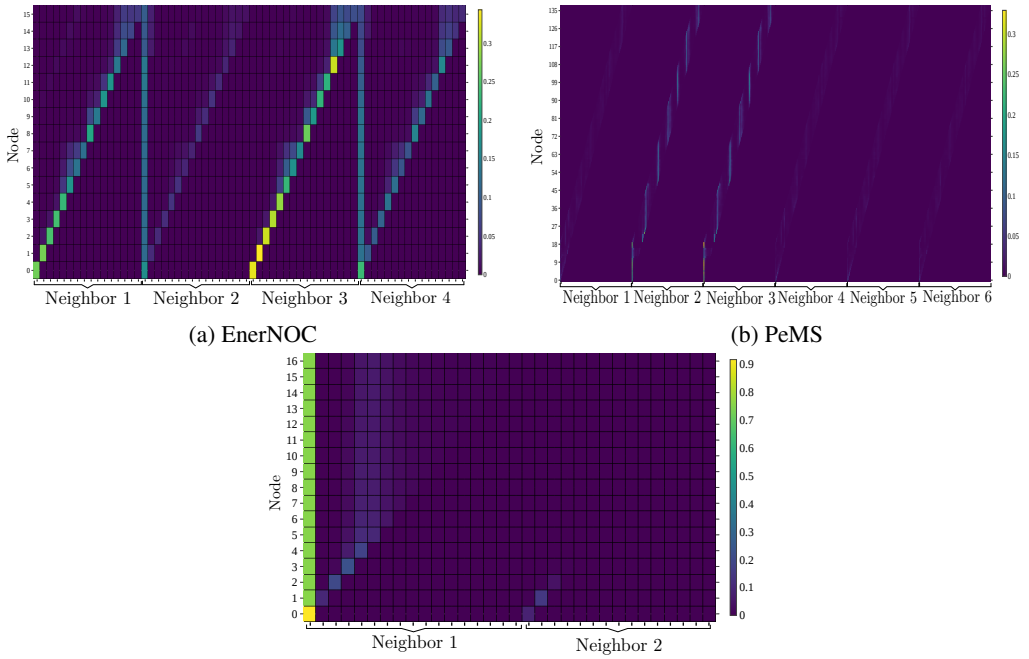
(c) Bank

Figure 7: Training and validation results for the real-world datasets. Note that MSE and RMSE are reported in a logarithmic scale.

Attention output Due to the proposed attention formulation, it is possible to interpret the graph structure detected. For example, from Fig. 8b it is possible to notice how the model detects Neighbor 2 and Neighbor 3 ad the most useful source of information to predict the traffic. A more in-depth investigation shows that such stations report the highest traffic flow, thus are the one that mostly influences the throughput of the target station.

It is also interesting to notice how the DIAN model is able to learn a meaningful graph structure in the EnerNOC dataset. Fig. 8a shows how erroneously we manually added four neighbours to a given node, while only three are relevant. As stated, such information is highly valuable because it enables the DIAN architecture to outperform methods based on handcrafted relations (Structural-RNN).

Finally, Fig. 8c demonstrate how no significant interactive structure is learned form the Bank dataset.



(a) EnerNOC

(b) PeMS

(c) Bank

Figure 8: Attention output of the DIAN model for the real-world datasets. Note that for space limitations we have not report neither the neighbours time series value, neither the node.